7. OSI-Schicht 4 – Transportschicht

- TCP (Transmission Control Protocol) Erklärung
- UDP (User Datagram Protocol) Erklärung
- TCP versus UDP

TCP (Transmission Control Protocol) Erklärung

TCP einfach erklärt

TCP steht für **Transmission Control Protocol**.

TCP ist ein **verbindungsorientiertes** und **zuverlässiges** Transportprotokoll. Es arbeitet auf der **Transportschicht** des TCP/IP-Modells und sorgt dafür, dass Daten möglichst vollständig, geordnet und fehlerfrei beim Empfänger ankommen.

Typische Anwendungen mit TCP sind zum Beispiel:

```
HTTP/HTTPS
SSH
E-Mail
Dateiübertragung
Remote Desktop
```

Grundidee von TCP

TCP wird verwendet, wenn Daten zuverlässig übertragen werden sollen.

Das bedeutet:

```
- vor der Datenübertragung wird eine Verbindung aufgebaut
- Daten werden nummeriert
- empfangene Daten werden bestätigt
- fehlende Daten können erneut übertragen werden
- Daten werden in der richtigen Reihenfolge an die Anwendung weitergegeben
```

TCP ist dadurch zuverlässiger als UDP, hat aber mehr Verwaltungsaufwand.

Merksatz:

```
TCP = verbindungsorientiert, zuverlässig und geordnet
UDP = verbindungslos, schnell und mit weniger Kontrolle
```

TCP arbeitet mit Ports

Eine IP-Adresse zeigt auf einen Host im Netzwerk.

Beispiel:

```
192.168.0.50
```

Ein Port zeigt auf einen bestimmten Dienst oder eine Anwendung auf diesem Host.

Beispiel:

```
192.168.0.50:443
```

Das bedeutet:

```
192.168.0.50 = Host / Gerät
443          = Dienst / Anwendung, hier HTTPS
```

Typische TCP-Ports:

```
80   = HTTP
443  = HTTPS
22   = SSH
25   = SMTP
110  = POP3
143  = IMAP
3389 = Remote Desktop
```

Merksatz:

```
IP-Adresse = welcher Host?
Port       = welcher Dienst?
Socket     = IP-Adresse + Port
```

Socket bei TCP

Ein Socket ist die Kombination aus IP-Adresse und Port.

Beispiel:

```
192.168.0.50:443
```

Bei einer TCP-Verbindung gibt es immer zwei Seiten:

```
Client-Socket: 192.168.0.20:51544
Server-Socket: 93.184.216.34:443
```

Der Client verwendet oft einen zufälligen hohen Quellport. Der Server verwendet meistens einen bekannten festen Zielport.

Eine vollständige TCP-Verbindung wird also durch diese Informationen beschrieben:

```
Quell-IP
Quell-Port
Ziel-IP
Ziel-Port
Protokoll TCP
```

TCP-Verbindungsaufbau: 3-Wege-Handshake

Bevor TCP Daten überträgt, wird eine Verbindung aufgebaut. Dieser Verbindungsaufbau heißt **3-Wege-Handshake**.

Dabei tauschen Client und Server Kontrollinformationen aus.

Ablauf:

```
Client                    Server

1. SYN  ---------------------->
   Verbindungswunsch

2. SYN-ACK <------------------
   Verbindungswunsch bestätigt

3. ACK  ---------------------->
   Bestätigung zurück
```

```
Verbindung steht
```

Was bedeutet SYN?

SYN ist die Abkürzung für:

```
Synchronize
```

Auf Deutsch:

```
synchronisieren
```

Beim TCP-Verbindungsaufbau bedeutet SYN:

```
Der Client möchte eine neue TCP-Verbindung starten
und seine Start-Sequenznummer mit dem Server synchronisieren.
```

SYN ist also nicht einfach nur „Hallo“, sondern enthält auch technische Informationen für den Start der Verbindung.

Was bedeutet SYN-ACK?

SYN-ACK besteht aus zwei Teilen:

```
SYN = Server möchte ebenfalls seine Start-Sequenznummer synchronisieren
ACK = Server bestätigt den SYN des Clients
```

Der Server sagt damit vereinfacht:

```
Ich habe deinen Verbindungswunsch erhalten.
Ich bin bereit.
Hier ist meine eigene Start-Sequenznummer.
```

Was bedeutet ACK?

ACK steht für:

```
Acknowledgement
```

Auf Deutsch:

```
Bestätigung
```

Beim dritten Schritt bestätigt der Client die Antwort des Servers.

Danach gilt:

```
Die TCP-Verbindung ist aufgebaut.
Daten können übertragen werden.
```

Merksatz:

```
SYN     = Verbindungswunsch + Start-Sequenznummer synchronisieren
SYN-ACK = Verbindungswunsch bestätigen + eigene Start-Sequenznummer senden
ACK     = Bestätigung zurücksenden
```

Sequenznummer ist keine Portnummer

Eine Sequenznummer ist nicht dasselbe wie eine Portnummer.

Eine **Portnummer** sagt:

```
Welcher Dienst oder welches Programm ist gemeint?
```

Eine **Sequenznummer** sagt:

```
An welcher Stelle im TCP-Datenstrom befinden sich diese Daten?
```

Merksatz:

```
Portnummer     = welcher Dienst?
Sequenznummer = welche Stelle im Datenstrom?
```

Oder einfacher:

Portnummer ist wie die Türnummer eines Dienstes.

Sequenznummer ist wie die Position der Daten im Datenstrom.

Was macht die Sequenznummer bei TCP?

TCP nummeriert nicht einfach nur einzelne Pakete wie „Paket 1, Paket 2, Paket 3“.

Technisch genauer:

TCP nummeriert die Bytes im Datenstrom.

Dadurch weiß der Empfänger:

- wo ein empfangenes TCP-Segment im Datenstrom beginnt
- ob Daten in der richtigen Reihenfolge angekommen sind
- ob Daten fehlen
- ob Daten doppelt angekommen sind

Ein TCP-Segment enthält eine Sequenznummer. Diese Sequenznummer zeigt, an welcher Position im Datenstrom die enthaltenen Daten beginnen.

Einfaches Beispiel für Sequenznummern

Segment 1: Sequenznummer 1000, enthält 500 Byte

Segment 2: Sequenznummer 1500, enthält 500 Byte

Segment 3: Sequenznummer 2000, enthält 500 Byte

Das bedeutet:

Segment 1 beginnt bei Byte 1000.

Segment 2 beginnt bei Byte 1500.

Segment 3 beginnt bei Byte 2000.

Wenn alles korrekt ankommt, kann der Empfänger die Daten in der richtigen Reihenfolge zusammensetzen.

Was passiert, wenn Daten verloren gehen?

Angenommen, Segment 2 geht verloren:

```
Segment 1 kommt an: Sequenznummer 1000
Segment 2 fehlt:    Sequenznummer 1500
Segment 3 kommt an: Sequenznummer 2000
```

Der Empfänger merkt:

```
Ich habe Daten ab 1000 bekommen.
Danach müsste eigentlich 1500 kommen.
Jetzt kommt aber schon 2000.
Also fehlt der Bereich ab 1500.
```

TCP erkennt dadurch, dass ein Teil der Daten fehlt.

Die fehlenden Daten können dann erneut übertragen werden.

ACK = Bestätigung empfangener Daten

Mit einem ACK bestätigt der Empfänger, welche Daten korrekt angekommen sind.

Vereinfacht gesagt:

```
ACK 1500 = Ich habe alles bis vor 1500 korrekt erhalten.
           Bitte sende als Nächstes die Daten ab 1500.
```

Wenn Daten fehlen, bestätigt der Empfänger nicht einfach alles als vollständig.

Stattdessen signalisiert er sinngemäß:

```
Ich erwarte weiterhin die Daten ab dieser Sequenznummer.
```

Dadurch erkennt der Sender, dass Daten erneut übertragen werden müssen.

Muss TCP komplett warten, wenn ein Segment fehlt?

Nicht unbedingt.

Später angekommene Daten können zwischengespeichert werden.

Beispiel:

```
Segment 1 kommt an.
Segment 2 fehlt.
Segment 3 kommt schon an.
```

TCP kann Segment 3 intern speichern.

Wichtig ist aber:

```
An die Anwendung werden die Daten erst in der richtigen Reihenfolge weitergegeben.
```

Das bedeutet:

```
Die Anwendung bekommt Segment 3 nicht vor Segment 2.
Erst wenn die fehlenden Daten angekommen sind,
werden die Daten geordnet weitergegeben.
```

Dadurch sieht die Anwendung einen zuverlässigen und geordneten Datenstrom.

Warum ist TCP zuverlässig?

TCP gilt als zuverlässig, weil es mehrere Kontrollmechanismen verwendet.

Dazu gehören:

```
- Verbindungsaufbau durch 3-Wege-Handshake
- Sequenznummern
- ACK-Bestätigungen
- erneute Übertragung verlorener Daten
- Reihenfolgekontrolle
- Erkennung doppelt empfangener Daten
- Flusskontrolle
```

Dadurch kann TCP sicherstellen, dass Daten vollständig und in der richtigen Reihenfolge beim Empfänger ankommen.

Flusskontrolle bei TCP

TCP verwendet Flusskontrolle, damit ein schneller Sender einen langsameren Empfänger nicht überfordert.

Der Empfänger kann dem Sender mitteilen, wie viele Daten er aktuell aufnehmen kann.

Das nennt man vereinfacht:

```
Empfangsfenster
```

Wenn der Empfänger nur wenig Speicher frei hat, kann er dem Sender signalisieren:

```
Sende langsamer oder warte kurz.
```

Dadurch wird verhindert, dass der Empfänger mit zu vielen Daten überlastet wird.

TCP-Verbindungsabbau

Eine TCP-Verbindung wird nicht einfach abrupt beendet, sondern kontrolliert geschlossen.

Dafür werden unter anderem FIN- und ACK-Nachrichten verwendet.

Vereinfacht:

```
Eine Seite sagt: Ich möchte die Verbindung beenden.
Die andere Seite bestätigt das.
Danach kann auch die andere Seite ihre Richtung schließen.
```

Typische Steuerbits beim Verbindungsabbau:

```
FIN = Verbindung geordnet beenden
ACK = Bestätigung
```

Merksatz:

```
SYN = Verbindung aufbauen
FIN = Verbindung beenden
ACK = bestätigen
```

TCP im Vergleich zu UDP

TCP:

```
- verbindungsorientiert
- zuverlässig
- geordnete Datenübertragung
- Bestätigungen durch ACKs
- erneute Übertragung verlorener Daten
- mehr Verwaltungsaufwand
```

UDP:

```
- verbindungslos
- geringer Verwaltungsaufwand
- keine eingebaute Zustellgarantie
- keine eingebaute Reihenfolgekontrolle
- keine eingebaute erneute Übertragung
- oft latenzärmer als TCP
```

TCP wird verwendet, wenn Zuverlässigkeit wichtig ist.

UDP wird häufig verwendet, wenn geringe Verzögerung wichtiger ist als vollständige Kontrolle.

Beispiel: TCP beim Webseitenaufruf

Wenn ein Client eine HTTPS-Webseite aufruft, passiert vereinfacht Folgendes:

```
1. Client möchte Website aufrufen.
2. Client baut TCP-Verbindung zum Server-Port 443 auf.
3. TCP führt den 3-Wege-Handshake aus.
4. Danach werden Daten übertragen.
5. TCP nummeriert die Daten mit Sequenznummern.
6. Der Empfänger bestätigt empfangene Daten mit ACKs.
7. Fehlende Daten werden erneut übertragen.
8. Die Anwendung erhält die Daten in der richtigen Reihenfolge.
```

Beispiel:

```
Client: 192.168.0.20:51544
Server: 93.184.216.34:443
```

```
Protokoll: TCP
```

Wichtige TCP-Begriffe

```
TCP = Transmission Control Protocol
SYN = Synchronize
ACK = Acknowledgement
FIN = Finish
Port = Dienstadresse auf einem Host
Socket = IP-Adresse + Port
Sequenznummer = Position der Daten im TCP-Datenstrom
3-Wege-Handshake = Verbindungsaufbau bei TCP
```

IHK-sichere Kurzformulierung

TCP ist ein verbindungsorientiertes und zuverlässiges Transportprotokoll. Vor der Datenübertragung wird durch den 3-Wege-Handshake eine Verbindung aufgebaut. TCP verwendet Sequenznummern, um die Reihenfolge der übertragenen Daten im Datenstrom festzulegen. Mit ACKs bestätigt der Empfänger korrekt erhaltene Daten. Fehlende Daten können erkannt und erneut übertragen werden. Dadurch stellt TCP eine geordnete und zuverlässige Datenübertragung bereit.

Merksätze

```
TCP = verbindungsorientiert, zuverlässig und geordnet
```

```
SYN     = Verbindung starten
SYN-ACK = Verbindung bestätigen und eigene Startnummer senden
ACK     = Bestätigung
```

```
Portnummer     = welcher Dienst?
Sequenznummer = welche Stelle im Datenstrom?
```

```
Sequenznummern helfen TCP zu erkennen,
ob Daten fehlen, doppelt angekommen sind
oder in falscher Reihenfolge eintreffen.
```

ACK bestätigt,

bis wohin Daten korrekt empfangen wurden.

Fehlende Daten werden erneut übertragen.

Später angekommene Daten können zwischengespeichert werden.

An die Anwendung geht alles erst in der richtigen Reihenfolge.

SYN = Verbindung aufbauen

FIN = Verbindung beenden

ACK = bestätigen

UDP (User Datagram Protocol) Erklärung

UDP einfach erklärt

UDP steht für **User Datagram Protocol**.

UDP ist ein **verbindungsloses** Transportprotokoll. Es arbeitet auf der **Transportschicht** des TCP/IP-Modells und wird verwendet, wenn Daten schnell und mit möglichst wenig Verwaltungsaufwand übertragen werden sollen.

UDP ist einfacher aufgebaut als TCP. Dafür bietet UDP aber keine eingebaute Garantie, dass Daten ankommen, vollständig sind oder in der richtigen Reihenfolge eintreffen.

Typische Anwendungen mit UDP sind zum Beispiel:

```
DNS
DHCP
VoIP
Video-Streaming
Audio-Streaming
Online-Gaming
QUIC / HTTP/3
```

Grundidee von UDP

UDP wird verwendet, wenn eine schnelle und einfache Datenübertragung wichtiger ist als vollständige Kontrolle.

Das bedeutet:

```
- keine feste Verbindung vor der Datenübertragung
- kein 3-Wege-Handshake
- keine eingebaute Zustellgarantie
- keine eingebaute Reihenfolgekontrolle
- keine automatische erneute Übertragung verlorener Daten
- geringer Verwaltungsaufwand
```

UDP sendet Daten einfach los. Ob die Gegenseite erreichbar ist oder ob die Daten vollständig ankommen, wird von UDP selbst nicht zuverlässig kontrolliert.

Merksatz:

```
UDP = verbindungslos, einfach und schnell
TCP = verbindungsorientiert, zuverlässig und geordnet
```

UDP arbeitet mit Ports

Auch UDP verwendet Ports, genau wie TCP.

Eine IP-Adresse zeigt auf einen Host im Netzwerk.

Beispiel:

```
192.168.0.50
```

Ein Port zeigt auf einen bestimmten Dienst oder eine Anwendung auf diesem Host.

Beispiel:

```
192.168.0.50:53
```

Das bedeutet:

```
192.168.0.50 = Host / Gerät
53           = Dienst / Anwendung, hier DNS
```

Typische UDP-Ports:

```
53    = DNS
67    = DHCP Server
68    = DHCP Client
123   = NTP
500   = IKE / IPsec
1194  = OpenVPN
51820 = WireGuard
443   = QUIC / HTTP/3
```

Wichtig:

```
Ein Port kann bei TCP und UDP unterschiedlich verwendet werden.
```

Beispiel:

```
TCP 443 = HTTPS über TCP
UDP 443 = QUIC / HTTP/3
```

Merksatz:

```
IP-Adresse = welcher Host?
Port       = welcher Dienst?
Socket     = IP-Adresse + Port
```

Socket bei UDP

Ein Socket ist die Kombination aus IP-Adresse und Port.

Beispiel:

```
192.168.0.50:53
```

Bei UDP kann ein Datenpaket von einem Quell-Socket zu einem Ziel-Socket gesendet werden.

Beispiel:

```
Client-Socket: 192.168.0.20:53000
Server-Socket: 192.168.0.1:53
```

Das bedeutet:

```
Client fragt von Port 53000 aus einen DNS-Server auf Port 53 an.
```

Eine UDP-Kommunikation wird also durch diese Informationen beschrieben:

```
Quell-IP
Quell-Port
Ziel-IP
Ziel-Port
Protokoll UDP
```

UDP hat keinen Verbindungsaufbau

Bei TCP gibt es vor der Datenübertragung einen 3-Wege-Handshake.

Bei UDP gibt es diesen Verbindungsaufbau nicht.

TCP:

```
Client                  Server

1. SYN  ---------------------->
2. SYN-ACK <------------------
3. ACK  ---------------------->

Verbindung steht
```

UDP:

```
Client                  Server

Datenpaket ------------------>

Keine vorherige Verbindung
Keine SYN-Nachricht
Kein SYN-ACK
Kein ACK durch UDP selbst
```

UDP sendet also direkt ein Datagramm an den Empfänger.

Merksatz:

```
TCP fragt vorher: Darf ich eine Verbindung aufbauen?
UDP sendet direkt los.
```

Was ist ein UDP-Datagramm?

Die Dateneinheit bei UDP nennt man **Datagramm**.

Ein UDP-Datagramm enthält unter anderem:

```
- Quellport
- Zielport
- Länge
- Prüfsumme
- Nutzdaten
```

UDP ist dadurch sehr schlank aufgebaut.

Wichtig:

```
UDP nummeriert die Daten nicht wie TCP mit Sequenznummern.
```

Deshalb kann UDP selbst nicht zuverlässig erkennen, ob ein Datagramm fehlt oder in falscher Reihenfolge angekommen ist.

UDP hat keine Sequenznummern wie TCP

TCP verwendet Sequenznummern, um die Position der Daten im Datenstrom festzulegen.

UDP macht das nicht.

TCP:

```
Segment 1: Sequenznummer 1000
Segment 2: Sequenznummer 1500
Segment 3: Sequenznummer 2000
```

UDP:

```
Datagramm 1 wird gesendet.
Datagramm 2 wird gesendet.
Datagramm 3 wird gesendet.

UDP selbst merkt sich keine Reihenfolge.
```

Wenn ein UDP-Datagramm verloren geht, wird es von UDP nicht automatisch erneut angefordert.

Wenn UDP-Datagramme in falscher Reihenfolge ankommen, sortiert UDP sie nicht automatisch.

Merksatz:

TCP kontrolliert die Reihenfolge.
UDP sendet einzelne Datagramme ohne Reihenfolgegarantie.

UDP hat keine eingebaute Zustellgarantie

UDP garantiert nicht, dass ein Datagramm beim Empfänger ankommt.

Ein UDP-Datagramm kann unterwegs verloren gehen, zum Beispiel durch:

- Netzwerküberlastung
- Paketverlust
- Routing-Probleme
- Firewall-Regeln
- fehlerhafte Übertragung

UDP selbst sendet verlorene Datagramme nicht automatisch erneut.

Das bedeutet:

Wenn ein UDP-Datagramm verloren geht,
merkt UDP selbst das nicht zuverlässig
und fordert es nicht automatisch erneut an.

Wenn eine Anwendung trotzdem Zuverlässigkeit braucht, muss sie diese selbst oberhalb von UDP einbauen.

Beispiele dafür sind:

- eigene Bestätigungen
- eigene Sequenznummern
- eigene Wiederholungen
- Fehlerkorrektur auf Anwendungsebene

UDP hat keine eingebaute Reihenfolgekontrolle

UDP garantiert nicht, dass Datagramme in der gesendeten Reihenfolge ankommen.

Beispiel:

```
Gesendet:
Datagramm 1
Datagramm 2
Datagramm 3

Angekommen:
Datagramm 1
Datagramm 3
Datagramm 2
```

UDP sortiert diese Datagramme nicht automatisch.

Wenn die Reihenfolge wichtig ist, muss die Anwendung selbst dafür sorgen.

Beispiel:

```
Eine Anwendung kann eigene Nummern in die Nutzdaten schreiben,
um die Reihenfolge später selbst zu prüfen.
```

Merksatz:

```
UDP liefert Datagramme so ab, wie sie ankommen.
UDP sortiert nicht automatisch.
```

UDP hat weniger Verwaltungsaufwand als TCP

UDP hat weniger Verwaltungsaufwand, weil es keinen Verbindungsaufbau, keine Sequenznummern, keine ACKs und keine automatische Wiederholung verlorener Daten gibt.

Dadurch ist UDP oft:

```
- einfacher
- schneller beim Start
- latenzärmer
- ressourcenschonender
```

Wichtig:

UDP ist nicht automatisch immer schneller im Sinne von höherer Datenrate.

Besser formuliert:

UDP hat weniger Verwaltungsaufwand als TCP und kann dadurch schneller bzw. latenzärmer sein.

Das ist besonders bei Anwendungen wichtig, bei denen Echtzeit wichtiger ist als perfekte Vollständigkeit.

Warum nutzt man UDP trotz fehlender Garantie?

UDP wird genutzt, weil bei manchen Anwendungen verlorene Daten weniger schlimm sind als Verzögerungen.

Beispiel VoIP:

Bei einem Telefonat ist es besser,
wenn ein kleines Audiostück kurz fehlt,
als wenn die Sprache stark verzögert ankommt.

Beispiel Online-Gaming:

Bei schnellen Positionsdaten ist es oft besser,
aktuelle Daten zu bekommen,
als alte verlorene Daten nachträglich zu übertragen.

Beispiel Streaming:

Bei Live-Video ist geringe Verzögerung wichtiger
als jedes einzelne Paket nachträglich zu retten.

Merksatz:

UDP wird häufig verwendet,
wenn Aktualität wichtiger ist als vollständige Nachlieferung.

Beispiel: DNS mit UDP

DNS verwendet sehr häufig UDP auf Port 53.

Ablauf vereinfacht:

```
1. Client möchte wissen, welche IP-Adresse zu einer Domain gehört.
2. Client sendet eine DNS-Anfrage per UDP an Port 53.
3. DNS-Server antwortet per UDP.
4. Die Antwort enthält die passende IP-Adresse.
```

Beispiel:

```
Client: 192.168.0.20:53000
DNS-Server: 192.168.0.1:53
Protokoll: UDP
```

Warum UDP hier sinnvoll ist:

```
- DNS-Anfragen sind meist klein
- eine Verbindung vorher aufzubauen wäre zusätzlicher Aufwand
- bei Verlust kann die Anfrage einfach erneut gestellt werden
```

Wichtig:

```
DNS kann auch TCP verwenden,
zum Beispiel bei größeren Antworten oder Zonentransfers.
```

Beispiel: VoIP mit UDP

Bei VoIP werden Sprachdaten in kleinen Paketen übertragen.

UDP eignet sich hier gut, weil geringe Verzögerung besonders wichtig ist.

Beispiel:

```
Sprecher → Sprachpakete → Netzwerk → Empfänger
```

Wenn ein kleines Sprachpaket verloren geht, ist das meist weniger schlimm als eine große Verzögerung.

Deshalb ist bei VoIP oft wichtiger:

```
geringe Latenz statt vollständige Nachlieferung
```

Beispiel: Online-Gaming mit UDP

Online-Spiele übertragen häufig Positionsdaten, Bewegungen und Zustände.

Beispiel:

```
Spielerposition
Blickrichtung
Bewegung
Aktionen
```

Wenn ein altes Positionspaket verloren geht, ist es oft nicht sinnvoll, es später noch nachzuliefern.

Wichtiger ist:

```
Der aktuelle Zustand soll möglichst schnell ankommen.
```

Deshalb wird für solche Echtzeitdaten häufig UDP genutzt.

QUIC und HTTP/3 nutzen UDP

QUIC ist ein modernes Transportprotokoll, das auf UDP basiert.

HTTP/3 verwendet QUIC.

Vereinfacht:

```
HTTP/3 läuft über QUIC.
QUIC läuft über UDP.
UDP läuft über IP.
```

Wichtig:

```
QUIC nutzt UDP als Grundlage,
baut aber eigene Funktionen für Zuverlässigkeit,
```

Verschlüsselung und Verbindungssteuerung ein.

Das bedeutet:

UDP selbst ist einfach und verbindungslos.
QUIC ergänzt darauf zusätzliche moderne Funktionen.

UDP und Firewall

Firewalls können UDP-Verkehr filtern, genau wie TCP-Verkehr.

Dabei werden zum Beispiel geprüft:

- Quell-IP
- Ziel-IP
- Quellport
- Zielport
- Protokoll UDP

Da UDP keine feste Verbindung wie TCP aufbaut, ist die Zustandsverfolgung schwieriger.

Eine Stateful Firewall kann sich aber trotzdem merken, dass ein interner Client ein UDP-Datagramm nach außen gesendet hat.

Beispiel:

Client sendet DNS-Anfrage an DNS-Server.
Firewall merkt sich diese Anfrage kurzzeitig.
DNS-Antwort darf zurück.
Unerwartete UDP-Pakete von außen werden blockiert.

Merksatz:

Auch UDP kann von einer Stateful Firewall verfolgt werden,
aber ohne echten TCP-Verbindungszustand.

UDP und NAT/PAT

Auch UDP kann über NAT/PAT ins Internet gehen.

Beispiel:

```
Interner Client: 192.168.0.20:53000
Öffentliche IP: 84.10.20.30:40001
DNS-Server: 1.1.1.1:53
```

Die Firewall bzw. der Router merkt sich die Zuordnung:

```
192.168.0.20:53000 → 84.10.20.30:40001
```

Wenn die Antwort vom DNS-Server zurückkommt, weiß der Router:

```
Diese Antwort gehört zurück an 192.168.0.20:53000.
```

Wichtig:

```
Bei UDP sind solche NAT-Zuordnungen meist zeitlich begrenzt,
weil es keine dauerhaft aufgebaute Verbindung wie bei TCP gibt.
```

UDP im Vergleich zu TCP

TCP:

```
- verbindungsorientiert
- 3-Wege-Handshake
- zuverlässig
- Sequenznummern
- ACK-Bestätigungen
- erneute Übertragung verlorener Daten
- Reihenfolgekontrolle
- mehr Verwaltungsaufwand
```

UDP:

```
- verbindungslos
- kein 3-Wege-Handshake
- keine eingebaute Zustellgarantie
```

- keine eingebaute Reihenfolgekontrolle
- keine automatische erneute Übertragung
- weniger Verwaltungsaufwand
- oft latenzärmer

Merksatz:

TCP kontrolliert stärker.
UDP ist schlanker und direkter.

Wann verwendet man TCP?

TCP verwendet man, wenn Daten zuverlässig und vollständig ankommen müssen.

Beispiele:

- Webseiten über HTTP/HTTPS
- Dateiübertragung
- SSH
- E-Mail
- Remote Desktop

Hier wäre es schlecht, wenn Daten fehlen oder in falscher Reihenfolge bei der Anwendung ankommen.

Wann verwendet man UDP?

UDP verwendet man häufig, wenn geringe Verzögerung wichtiger ist als vollständige Kontrolle.

Beispiele:

- DNS
- DHCP
- VoIP
- Live-Streaming
- Online-Gaming
- NTP
- VPN-Protokolle wie WireGuard
- QUIC / HTTP/3

Hier ist es oft besser, schnell weiterzumachen, statt verlorene alte Daten nachzuliefern.

Ist UDP unsicherer als TCP?

UDP ist nicht automatisch „unsicherer“ als TCP.

UDP hat nur weniger eingebaute Kontrollfunktionen.

Sicherheit hängt vor allem davon ab:

```
- welches Anwendungsprotokoll verwendet wird
- ob Verschlüsselung eingesetzt wird
- wie die Firewall konfiguriert ist
- ob der Dienst korrekt abgesichert ist
```

Beispiel:

```
QUIC nutzt UDP,
kann aber trotzdem verschlüsselte Kommunikation ermöglichen.
```

UDP bedeutet also nicht automatisch unsicher. Es bedeutet nur:

```
UDP selbst garantiert weniger als TCP.
```

Wichtige UDP-Begriffe

```
UDP = User Datagram Protocol
Datagramm = einzelne UDP-Dateneinheit
Port = Dienstadresse auf einem Host
Socket = IP-Adresse + Port
Quellport = Port des sendenden Systems
Zielport = Port des empfangenden Dienstes
Prüfsumme = einfache Fehlererkennung im UDP-Datagramm
```

IHK-sichere Kurzformulierung

UDP ist ein verbindungsloses Transportprotokoll mit geringem Verwaltungsaufwand. Im Gegensatz zu TCP baut UDP vor der Datenübertragung keine Verbindung auf und bietet keine eingebaute

Garantie für Zustellung, Reihenfolge oder erneute Übertragung verlorener Daten. Dadurch ist UDP besonders für Anwendungen geeignet, bei denen geringe Verzögerung wichtiger ist als vollständige Kontrolle, zum Beispiel DNS, VoIP, Streaming oder Online-Gaming.

Merksätze

UDP = verbindungslos, einfach und mit wenig Verwaltungsaufwand

UDP sendet direkt los.
TCP baut vorher eine Verbindung auf.

UDP garantiert nicht,
dass Datagramme ankommen,
in richtiger Reihenfolge ankommen
oder erneut übertragen werden.

UDP ist oft latenzärmer als TCP,
aber nicht automatisch immer schneller in jeder Situation.

TCP = Zuverlässigkeit und Kontrolle
UDP = Geschwindigkeit und geringer Verwaltungsaufwand

Wenn UDP Zuverlässigkeit braucht,
muss die Anwendung diese selbst einbauen.

DNS, VoIP, Streaming und Online-Gaming
sind typische Beispiele für UDP.

TCP versus UDP

TCP und UDP sind beide Transportprotokolle. Sie arbeiten auf der Transportschicht und sorgen dafür, dass Daten zwischen Anwendungen auf verschiedenen Geräten übertragen werden können.

Der wichtigste Unterschied ist:

```
TCP baut vor der Datenübertragung eine Verbindung auf.
UDP sendet Daten ohne vorherigen Verbindungsaufbau direkt los.
```

TCP kurz erklärt

TCP steht für **Transmission Control Protocol**.

TCP ist:

```
- verbindungsorientiert
- zuverlässig
- geordnet
- kontrolliert
- mit mehr Verwaltungsaufwand verbunden
```

Bei TCP wird vor der Übertragung der eigentlichen Nutzdaten zuerst eine Verbindung aufgebaut. Dieser Verbindungsaufbau heißt **3-Wege-Handshake**.

```
Client                    Server

1. SYN  ---------------------->   Verbindungswunsch
2. SYN-ACK <------------------   Bestätigung + eigene Start-Sequenznummer
3. ACK  ---------------------->   Bestätigung

Danach ist die TCP-Verbindung aufgebaut.
Erst danach werden die eigentlichen Nutzdaten übertragen.
```

Wichtig:

```
Nicht ACK ist der erste Schritt.
Der erste Schritt ist SYN.
```

```
ACK ist die Bestätigung im dritten Schritt.
```

Was bedeutet SYN bei TCP?

SYN steht für:

```
Synchronize
```

Auf Deutsch:

```
synchronisieren
```

SYN bedeutet beim TCP-Verbindungsaufbau:

```
Der Client möchte eine neue TCP-Verbindung starten
und seine Start-Sequenznummer mit dem Server synchronisieren.
```

SYN ist also der Verbindungswunsch und gleichzeitig der Start der Synchronisation der Sequenznummern.

Was bedeutet SYN-ACK bei TCP?

SYN-ACK besteht aus zwei Teilen:

```
SYN = Server möchte ebenfalls seine Start-Sequenznummer synchronisieren
ACK = Server bestätigt den SYN des Clients
```

Der Server sagt damit vereinfacht:

```
Ich habe deinen Verbindungswunsch erhalten.
Ich bin bereit.
Hier ist meine eigene Start-Sequenznummer.
```

Was bedeutet ACK bei TCP?

ACK steht für:

```
Acknowledgement
```

Auf Deutsch:

```
Bestätigung
```

Mit ACK bestätigt eine Seite, dass bestimmte Daten oder Steuerinformationen angekommen sind.

Beim 3-Wege-Handshake bestätigt der Client mit ACK die Antwort des Servers.

Merksatz:

```
SYN     = Verbindung starten und Sequenznummer synchronisieren
SYN-ACK = Verbindung bestätigen und eigene Startnummer senden
ACK     = Bestätigung zurücksenden
```

UDP kurz erklärt

UDP steht für **User Datagram Protocol**.

UDP ist:

```
- verbindungslos
- einfach
- schnell bzw. oft latenzärmer
- mit wenig Verwaltungsaufwand verbunden
- ohne eingebaute Zustellgarantie
- ohne eingebaute Reihenfolgekontrolle
- ohne automatische erneute Übertragung verlorener Daten
```

Bei UDP gibt es keinen 3-Wege-Handshake.

```
Client                  Server

Daten ---------------------->

Kein SYN
Kein SYN-ACK
```

```
Kein verbindungsaufbauendes ACK
```

UDP sendet Datagramme direkt los. UDP selbst prüft nicht zuverlässig, ob die Gegenseite bereit ist oder ob die Daten vollständig angekommen sind.

Nutzdaten bei TCP und UDP

Nutzdaten sind die eigentlichen Inhalte, die eine Anwendung übertragen möchte.

Beispiele:

```
- Webseiteninhalte
- Dateien
- Sprache
- Videodaten
- DNS-Anfragen
- Spielinformationen
```

Bei TCP gilt:

```
Erst Verbindungsaufbau,
dann Nutzdaten.
```

Bei UDP gilt:

```
Kein Verbindungsaufbau,
Nutzdaten werden direkt als Datagramm gesendet.
```

Sequenznummern bei TCP

TCP verwendet Sequenznummern.

Eine Sequenznummer ist keine Portnummer.

```
Portnummer     = welcher Dienst?
Sequenznummer = welche Stelle im Datenstrom?
```

TCP nummeriert technisch gesehen die Bytes im Datenstrom. Dadurch kann der Empfänger erkennen:

```
- welche Daten angekommen sind
- ob Daten fehlen
- ob Daten doppelt angekommen sind
- ob Daten in falscher Reihenfolge angekommen sind
```

Beispiel:

```
Segment 1: Sequenznummer 1000, enthält 500 Byte
Segment 2: Sequenznummer 1500, enthält 500 Byte
Segment 3: Sequenznummer 2000, enthält 500 Byte
```

Wenn Segment 2 fehlt, merkt TCP:

```
Nach 1000 müsste 1500 kommen.
Wenn schon 2000 kommt, fehlt der Bereich ab 1500.
```

Fehlende Daten können dann erneut übertragen werden.

UDP hat keine TCP-Sequenznummern

UDP nummeriert die Daten nicht wie TCP mit Sequenznummern.

UDP sendet einzelne Datagramme.

```
Datagramm 1 wird gesendet.
Datagramm 2 wird gesendet.
Datagramm 3 wird gesendet.
```

UDP selbst merkt sich dabei keine Reihenfolge.

Wenn ein Datagramm verloren geht, fordert UDP es nicht automatisch erneut an.

Wenn Datagramme in falscher Reihenfolge ankommen, sortiert UDP sie nicht automatisch.

Zuverlässigkeit

TCP ist zuverlässig, weil es mehrere Kontrollmechanismen verwendet:

- Verbindungsaufbau durch 3-Wege-Handshake
- Sequenznummern
- ACK-Bestätigungen
- erneute Übertragung verlorener Daten
- Reihenfolgekontrolle
- Erkennung doppelt empfangener Daten
- Flusskontrolle

UDP hat diese Zuverlässigkeit nicht eingebaut.

UDP garantiert nicht:

- dass Daten ankommen
- dass Daten vollständig ankommen
- dass Daten in der richtigen Reihenfolge ankommen
- dass verlorene Daten erneut übertragen werden

Wenn eine Anwendung über UDP trotzdem Zuverlässigkeit benötigt, muss sie diese selbst einbauen.

Beispiele:

- eigene Bestätigungen
- eigene Sequenznummern
- eigene Wiederholungen
- Fehlerkorrektur auf Anwendungsebene

Geschwindigkeit und Verwaltungsaufwand

TCP hat mehr Verwaltungsaufwand, weil es Verbindungen aufbaut, Daten bestätigt, Reihenfolgen prüft und verlorene Daten erneut überträgt.

UDP hat weniger Verwaltungsaufwand, weil es diese Funktionen nicht eingebaut hat.

Deshalb ist UDP oft latenzärmer.

Wichtig:

UDP ist nicht automatisch immer schneller im Sinne von höherer Datenrate.
UDP hat aber weniger Verwaltungsaufwand und kann dadurch schneller reagieren.

Besser formuliert:

```
UDP ist oft latenzärmer als TCP,
weil es keinen Verbindungsaufbau und weniger Kontrollmechanismen hat.
```

Typische Anwendungen für TCP

TCP wird verwendet, wenn Daten zuverlässig und vollständig ankommen müssen.

Beispiele:

```
- HTTP/HTTPS über TCP
- SSH
- E-Mail
- Dateiübertragung
- Remote Desktop
- Datenbankverbindungen
```

Hier wäre es problematisch, wenn Daten fehlen oder in falscher Reihenfolge bei der Anwendung ankommen.

Beispiel:

```
Bei einer Dateiübertragung darf kein Teil der Datei fehlen.
Bei SSH müssen Befehle korrekt und in richtiger Reihenfolge ankommen.
Bei Webseiten sollen Inhalte vollständig übertragen werden.
```

Typische Anwendungen für UDP

UDP wird häufig verwendet, wenn geringe Verzögerung wichtiger ist als vollständige Kontrolle.

Beispiele:

```
- DNS
- DHCP
- VoIP
- Live-Streaming
- Online-Gaming
- NTP
```

```
- WireGuard
- QUIC / HTTP/3
```

Beispiel VoIP:

```
Bei einem Telefonat ist es besser,
wenn ein kleines Audiostück kurz fehlt,
als wenn die Sprache stark verzögert ankommt.
```

Beispiel Online-Gaming:

```
Bei schnellen Positionsdaten ist der aktuelle Zustand wichtiger
als ein altes verlorenes Paket nachträglich zu übertragen.
```

DNS als Beispiel für UDP

DNS nutzt sehr häufig UDP auf Port 53.

Ablauf vereinfacht:

```
1. Client fragt per UDP beim DNS-Server an.
2. DNS-Server antwortet per UDP.
3. Die Antwort enthält die passende IP-Adresse zur Domain.
```

Beispiel:

```
Client:     192.168.0.20:53000
DNS-Server: 192.168.0.1:53
Protokoll:  UDP
```

Warum UDP hier sinnvoll ist:

```
- DNS-Anfragen sind meist klein.
- Ein TCP-Verbindungsaufbau wäre zusätzlicher Aufwand.
- Bei Verlust kann die Anfrage einfach erneut gestellt werden.
```

Wichtig:

DNS kann auch TCP verwenden,
zum Beispiel bei größeren Antworten oder Zonentransfers.

QUIC und HTTP/3

QUIC ist ein modernes Transportprotokoll, das auf UDP basiert.

HTTP/3 läuft über QUIC.
QUIC läuft über UDP.
UDP läuft über IP.

Wichtig:

UDP selbst ist einfach und verbindungslos.
QUIC baut darauf eigene Funktionen für Verbindung, Zuverlässigkeit und Verschlüsselung auf.

Das bedeutet:

Nur weil UDP selbst keine TCP-Zuverlässigkeit bietet,
kann ein Protokoll oberhalb von UDP trotzdem eigene Kontrollmechanismen einbauen.

TCP und UDP mit Ports

Sowohl TCP als auch UDP verwenden Ports.

IP-Adresse = welcher Host?
Port = welcher Dienst?
Socket = IP-Adresse + Port

Wichtig:

TCP-Port 443 und UDP-Port 443 sind technisch getrennt.

Beispiel:

```
TCP 443 = HTTPS über TCP
UDP 443 = QUIC / HTTP/3
```

Ein Dienst kann also denselben Port bei TCP und UDP unterschiedlich verwenden.

Firewall-Bezug

Firewalls können sowohl TCP als auch UDP filtern.

Dabei prüfen sie zum Beispiel:

```
- Quell-IP
- Ziel-IP
- Quellport
- Zielport
- Protokoll TCP oder UDP
```

Bei TCP kann eine Stateful Firewall den Verbindungszustand gut erkennen:

```
SYN → SYN-ACK → ACK → Verbindung steht
```

Bei UDP gibt es keinen echten Verbindungszustand wie bei TCP. Eine Stateful Firewall kann sich aber trotzdem kurzzeitig merken, dass ein internes Gerät ein UDP-Datagramm nach außen gesendet hat.

Beispiel:

```
Client sendet DNS-Anfrage per UDP nach außen.
Firewall merkt sich diese Anfrage kurzzeitig.
DNS-Antwort darf zurück.
Unerwartete UDP-Pakete von außen werden blockiert.
```

TCP versus UDP als Tabelle

Merkmal	TCP	UDP
Voller Name	Transmission Control Protocol	User Datagram Protocol
Verbindungsaufbau	Ja, 3-Wege-Handshake	Nein
Erste Nachricht	SYN	Direkt Datagramm/Nutzdaten

Merkmal	TCP	UDP
Zuverlässigkeit	Eingebaut	Nicht eingebaut
Reihenfolgekontrolle	Ja	Nein
Sequenznummern	Ja	Nicht wie TCP
ACK-Bestätigungen	Ja	Nicht durch UDP selbst
Erneute Übertragung	Ja	Nein
Verwaltungsaufwand	Höher	Geringer
Latenz	Oft höher	Oft geringer
Typische Nutzung	Zuverlässige Datenübertragung	Echtzeit oder kleine schnelle Anfragen
Beispiele	HTTPS, SSH, E-Mail, Dateiübertragung	DNS, VoIP, Streaming, Gaming, QUIC

Einfacher Vergleich

TCP ist wie ein Einschreiben:

```
Es wird geprüft, bestätigt und bei Problemen erneut gesendet.
```

UDP ist wie eine Postkarte:

```
Sie wird direkt abgeschickt, aber es gibt keine eingebaute Garantie,
dass sie ankommt oder in welcher Reihenfolge sie ankommt.
```

IHK-sichere Kurzformulierung

TCP ist ein verbindungsorientiertes und zuverlässiges Transportprotokoll. Vor der Datenübertragung wird über den 3-Wege-Handshake eine Verbindung aufgebaut. TCP verwendet Sequenznummern, ACK-Bestätigungen, Reihenfolgekontrolle und erneute Übertragung verlorener Daten. Dadurch eignet sich TCP für Anwendungen, bei denen Daten vollständig und korrekt ankommen müssen.

UDP ist ein verbindungsloses Transportprotokoll mit geringem Verwaltungsaufwand. UDP baut vor dem Senden keine Verbindung auf und bietet keine eingebaute Garantie für Zustellung, Reihenfolge oder erneute Übertragung verlorener Daten. Dadurch eignet sich UDP besonders für Anwendungen, bei denen geringe Verzögerung wichtiger ist als vollständige Kontrolle.

Merksätze

TCP = erst Verbindung aufbauen, dann Nutzdaten senden

UDP = keine Verbindung aufbauen, Daten direkt senden

TCP = zuverlässig, geordnet und kontrolliert

UDP = verbindungslos, schlank und oft latenzärmer

TCP fragt vorher:
"Darf ich eine Verbindung aufbauen?"

UDP sendet direkt:
"Hier sind die Daten."

TCP nutzt SYN, SYN-ACK und ACK für den Verbindungsaufbau.

UDP hat keinen 3-Wege-Handshake.

TCP erkennt fehlende Daten und überträgt sie erneut.

UDP macht das nicht automatisch.

TCP eignet sich für vollständige Daten.

UDP eignet sich für schnelle oder zeitkritische Daten.